1 PAGINATION

1. Tampilan home :
   a. di fungsi index() pada BlogController:
      `$blogs = Blog:all();` //all kan nampilin semua data
      //ganti dengan ;
      `$blogs = Blog:paginate(5);`
      //paginate nampilin bayak data sesuai parameter yang diberikan
   b. di view homenya yaitu di home.blade.php tambahin:
      1. `{{$blogs->links()}}` //atau `{{blogs->render()}}`
         //buat << 1 2 3 4 >>
      2. `jika uri nya ada parameter lain, misal:`
         localhost:8000/blog?cat=satu&page=5
         semua parameter lain akan hilang jika dipindah ke page lain, biar nggk hilang beri tambahan:
         `{{ $blogs ->appends(Request::input())->links() }}`
2. Tampilan << 1 2 3 4 >> bisa diedit karena laravel secara otomatis akan membuat ini dengan class bernama pagination

2 RESOURCE CONTROLLER

1. Guna; memudahkan kita membuat create, update, delete, dengan 1 routes dan command
2. Langkah :
   a. `php artisan make:controller PhotoController --resource`
      --> secara otomatis di PhotoController akan membuat fungsi: index(), create(), store(Request $request), show(), edit(), destroy()
   b. untuk mendaftarkan routesnya:
      `Route::resource('photos', 'PhotoController');`
      //urinya //controllernya
   --> Route tersebut akan menghasilkan : meskipun nggk tertulis:

Verb	URI	Action	Route Name
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

   --> untuk liat routesnya yang nggk tertulis:
   `php artisan route:list`

3 MORE ABOUT ROUTES

1. `Route::match(['get','post'], 'blog/testing', 'BlogController@testing');`
   --> Cara guna:
   1. di controller :pake if..else.../ switch:

```
public function testing (Request $request)
{
  if($request->method()=="GET")   //untuk GET
  {
    ..............//beri perintah jika get yg terjadi
  }
  else //untuk POST
  {
    ..............//beri perintah jika post yg terjadi
  }
```

```
    }
```

2. Memberi nama route:
   a. `Route::get('/blog', 'BlogController@index')->name('coba');` //ini
   b. Cara gunain:
      `return redirect()->route('coba');` //sesuai nama routenya
3. Prefix
   a. Jika di route ada kata sama yang banyak (`exp:/blog`) kata tersebut bisa disingkat dengan:

```
Route::group(['prefix'=>'blog'], function()
{
 Route::get('/', 'BlogController@index');
 Route::get('/create', 'BlogController@create');
   //semua kata 'blog' pada route bisa dihilangkan
}
);
```

4 SYSTEM AUTH 101

1. `php artisan make:auth`
   → `guna system autentikasi(login dan register) bawaan dari laravel`
2. yang terbentuk dari command ini :
   a. di Controller:
      1. HomeController.php //untuk page setelah berhasil login
      2. di Auth directory:
         1. ForgotPasswordController.php
         2. LoginController.php
         3. RegisterController.php
         4. ResetPasswordController.php
   b. di views:
      1. home.blade.php
      2. auth directory:
         1. register.blade.php --> berisi halaman register dengan template bootstrap
         2. login.blade.php --> berisi halaman login dengan template bootstrap
      3. layout directory:
         app.blade.php //authetikasi apakah dia guest/user disini
   c. di routes:
      1. `Auth::routes();`
         `Routes::get('/home', 'HomeController@index');`
3. Langkah2:
   1. `php artisan make:auth`
   2. setting `.env` untuk databasenya:`DB_DATABASE, DB_HOST, etc`
   3. `php artisan migrate`
      `-->` langsung bisa karena sebelumnya di migration dir sudah ada file migration untuk table users dan reset_password

5 CONTOH HALAMAN PROFILE

1. di LoginController.php :
   ada `var $reditecTo = '/home';` //yang berisi jika berhasil login akan redirect ke /home, bisa diganti ke redirect lain
2. di HomeController.php ada:

```
  public function __construct()
   {
       $this->middleware('auth');
   }
```

   `-->` fungsi __construct akan dipanggil secara otomatis jika controller dipanggil
   `-->` `$this->middleware('auth');` berguna untuk authentikasi (boleh masuk nggk nya)
   // paling penting disini.
3. middleware bisa ditaruh langsung di route:
   `Route::get('/profile','UserController@profie')->middleware('auth');` //ini
4. Cara ngambil data user di database yang memakai system authentikasi laravel:
   a. di Controller yang ingin memakai:
      1. `use Illuminate\Support\Facades\Auth;`
      2. di fungsi:

```
public function profile()
{
 $auth = Auth::user()->email; //untuk ngambil email user
}
```

6 CARA UPLOAD GAMBAR

1. Langkah2:
   a. di viewnya (create.blade.php) beri:
      --> `<input type="file" name="featured_image">`
      --> di form beri tambahan:
      `enctype="multipart/form_data"`
   b. di controller di fungsi store():
      `$request->file('featured_image')->store('simpan_sini')`
      //namenya dari form //dir tempat nyimpannya
      --> file gambar akan tersimpan di `storage/app/nama_dir_nyimpannya`
      --> cara nyimpan gambar dengan nama ttn:
      ```
      $beri_nama_gambarnya = time().'.png';
      $request->file('featured_image')
              ->storeAs('dir_simpannya','$beri_nama_gambarnya');
      ```
      //masukin nama ke database
      `$blog->featured_image = $beri_nama_gambarnya;`
      --> validasi:
      ```
      1. $this->validate($request,
          [
            'featured_image' =>'mimes:jpeg,jpg,png|max:10000'
          ]);
      ```
      2. di view beri tahu jika error

7 MENGELUARKAN GAMBAR DARI STORAGE

1. Yang harus dipahami, secara default file gambar akan tersimpan di storage/app/nama_dirnya, agar bisa mengaksesnya kita harus membuat visibilitynya jadi public, caranya:
   a. di controller:
   ```
   $request->file('featured_image')
    ->storeAs('public/dir_simpannya','$beri_nama_gambarnya');
   //liat: taro di storage/app/public/...
   ```
   b. beri akses dari dir public ke storage`/app/public` nya dengan cara:
   `php artisan storage:link`
   c. Lalu cara aksesnya dari view:
   ```
   <image src=" {{asset('storage/nama_dir_nyimpannya/'blog->featured_image)}} " alt="">
   ```
   //nama_tablenya->coloum_gambarnya

8 MENGIRIM EMAIL

1. Menggunakan mailtrap.io buat ngetest emailnya beneran kekirim apa nggk
2. Langkah-langkah konfigurasi awal :
   1. register ke mailtrap.io --> setelah login -> klik ke Demo Inbox akan muncul crendentials yang akan dimasukkan ke .env
   2. setting config/mail di .env. dari
      - MAIL_DRIVER=smtp
      - MAIL_HOST=mailtrap.io
      - MAIL_PORT=2525
      - MAIL_USERNAME= isi sesuai credentials yang didapat di Demo Inbox di bagian SMTP
      - MAIL_PASSWORD= isi sesuai credentials yang didapat di Demo Inbox di bagian SMTP
3. Pengaplikasian : kasus : user ketika post suatu artikel, kita akan ngasih notif email “post telah dikirim” :
   1. `php artisan make:mail BlogPosted`
   --> akan terbentuk di \App dir \Mail yang berisi BlogPosted.php
   --> di BlogPosted terdapat fungsi __construct() dan build()
   --> Fungsi build() ni adalah fungsi utama untuk dipakai mengirim emailnya
   --> Cara menggunakan : panggil BlogPosted dimana kita mau ngirim emailnya
   2. exp :
      a) di store blog di BlogController:
      ```
      $blog  = new Blog;
      $blog->title = $request->title;
      $blog->description = $request->description;
      $blog->save();

      //sebelum redirect kita kirim pemberitahuan lewat email kalau artikel berhasil di post
      Mail::to('test@emailuser.com')->send(new BlogPosted());
              //email user          //file emailnya

      return redirect('/blog');
      ```

b) di BlogController :

```
use Illuminate\Support\Facades\Mail;
use App\Mail\BlogPosted;
```

c) di views bikin blade di dir \emails dengan nama blog.blade.php, yang berguna sebagai isi pesan di email :

```
<h1>Post berhasil dibuat</h1>
```

d) di BlogPosted.php isi fungsi build() dengan :

```
public function build(){
        return $this->from('admin@admin.com')view('emails.blog');
                                //email adminnya          //isi emailnya
}
```

--> email admin bisa di setting juga di app\config\mail di 'from'

4. Tambahan jika pengen masukin parameter di eisi emailnya menjadi "post telah dikirim dengan judul":
   1. `Mail::to('test@emailuser.com')->send(new BlogPosted($blog));`
      //oper $blog ke BlogPosted
   2. di BlogPosted isi dengan :

```
use App\Models\Blog;

public $blog;
public function __construct(Blog $blog){
        $this->blog = $blog;
}
```

   3. di view di emails.blog : kita bisa pasing $blog tanpa harus ngasih parameter di buildnya:

```
<h1>Post berhasil dibuat dengan judul {{$blog->title}}</h1>
```

9 MIDDLEWARE ADMIN

1. idenya adalah membuat 3 role, yaitu guest, user,dan admin dengan menaruh role di database
2. Langkah2 awal:
   a. Setting database, `php artisan make:auth`
   b. ini penting:
      --> sebelum migrate, beri tambahan pada create_user_table.php pada folder migration:

```
$table->tinyInteger('role')->default('1');
```

      //ini berguna untuk role, misal jika role 1 dia user dan jika role 2 dia admin
   c. `php artisan migrate`
3. Langkah2 lanjutan (untuk admin):
   a. di routes tambah 1 route untuk admin:

```
Route::get('/admin', 'AdminController@dashboard');
```

   b. Buat AdminController dan buat fungsi dashboard() yang menuju halaman dashboard
   c. Buat middleware admin:

```
php artisan make:middleware AdminMiddleware
```

   d. di AdminMiddleware:

```
pubic functionhandle($request, Closure $next){
      $user = $request->user();
      if($user){
       //jika user ada (bukan guest) which means udah login
            if ($user->isAdmin()){
                  return $next($request);
                  //jika dia admin akan next ke function selanjutnya
            }
      }
      return abort (403);
                  //jika dia user atau guest dan bukan admin, akan ada error 403 jika mencoba akses }
```

   e. di Model User.php:
      --> beri keterangan tentang fungsi isAdmin() diatas:

```
  public function isAdmin(){
      if (this->role == 2) return true;
      return false;
  }
```

   f. Untuk mendaftarkan AdminMiddleware ke route, kita perlu mendaftarkannya di Kernel.php dalam $routeMiddleware:

```
'admin' => \App\Http\Middleware\AdminMiddleware::class;
```

   g. Nah dari nama 'admin' ini kita bisa memakainya untuk middleware di route yang diinginkan, misal di bagian dashboard:

```
Route::group(['middleware'] => 'admin', function(){
 Route::get('/admin','adminController@dashboard');
});
```